UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/708,159	11/08/2000	Toshiaki Yasue	JP919990097US1	1032

7590 07/09/2007

William A Kinnaman Jr
IBM Corporation - MS P386
2455 South Road
Poughkeepsie, NY 12601

EXAMINER	
RUTTEN, JAMES D	

ART UNIT	PAPER NUMBER
2192	

MAIL DATE	DELIVERY MODE
07/09/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Application Number: 09/708,159
Filing Date: November 08, 2000
Appellant(s): YASUE ET AL.

William A. Kinnaman, Jr., Reg. No. 27,650
For Appellant

EXAMINER'S ANSWER

This is in response to the appeal brief filed 2/12/07 appealing from the Office action mailed 8/11/06.

(1) Real Party in Interest

A statement identifying by name the real party in interest is contained in the brief.

(2) Related Appeals and Interferences

The examiner is not aware of any related appeals, interferences, or judicial proceedings which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

(3) Status of Claims

The statement of the status of claims contained in the brief is correct.

(4) Status of Amendments After Final

The appellant's statement of the status of amendments after final rejection contained in the brief is correct.

(5) Summary of Claimed Subject Matter

The summary of claimed subject matter contained in the brief is correct.

(6) Grounds of Rejection to be Reviewed on Appeal

The appellant's statement of the grounds of rejection to be reviewed on appeal is correct.

(7) Claims Appendix

The copy of the appealed claims contained in the Appendix to the brief is correct.

(8) Evidence Relied Upon

6,513,156	BAK	01-2003
5,530,866	KOBLENZ ET AL	06-1996

Aho et al. "Compilers: Principles, Techniques, and Tools", Addison-Wesley, 1986, Chapter 10

Bacon et al. "Compiler Transformations for High-Performance Computing" ACM Computing Surveys, Vol. 26, Issue 4, (Dec 1994) pp. 345-420

(9) Grounds of Rejection

The following grounds of rejection are duplicated from the 8/11/06 Final Office Action, and are applicable to the appealed claims:

- Claims 1-4 are rejected under 35 U.S.C. 103(a) as being unpatentable over prior art of record "Compilers: Principles, Techniques, and Tools" by Aho et al. (hereinafter "Aho") in view of prior art of record U.S. Patent 6,513,156 to Bak et al. (hereinafter referred to as "Bak"), further in view of prior art of record "Compiler Transformations for High-Performance Computing" by Bacon et al. (hereinafter referred to as "Bacon"), further in view of U.S. Patent 5,530,866 to Koblenz et al. (hereinafter "Koblenz").

In regard to claim 1, Aho discloses:

A program execution method ... comprising the steps of: optimizing the loop process, said optimizing step including the steps of: copying code from the top of the loop process to a point that post-dominates said top of said loop process and said one or more ...points to a location immediately preceding said loop process if said ... points are located inside said loop process; Aho discusses code optimizations that benefit from the reducibility of flow graphs containing loops in terms of using intervals, interval graphs, and node splitting (see pages 664-668). In particular, node splitting is a technique that is used to produce a limit flow graph of a single node (page 666, last paragraph). Additional nodes are created that precede the original node. Fig. 10.49 shows a flow graph with 3 nodes with edges from 1 to 2, 1 to 3, 2 to 3, and 3 to 2. The loop of nodes 2 and 3 show two incoming edges. Node 2 is split into nodes 2a and 2b. When combined with node 1, node 2a now precedes the loop that is formed between nodes 2b and 3. Further description is found on pages 679-680, which describes duplicating a region that represents a node, and placing that region in a location preceding the node. This process is shown in Fig. 10.57 on page 680.

Aho further discloses common subexpression elimination (Fig. 10.7 on page 593), and code motion (page 596). Also, as asserted by Applicant (see 4/1/04, page 7), Aho discloses the notion of transfer points (Section 10.4, pages 602-608).

Aho does not expressly disclose: moving transfer points to the top of a loop process; transferring a method from an interpreter process to a compiled code process; storing information for generating recalculation code for specific transfer points; performing a recalculation during a transfer process; or privatization.

However, in an analogous environment, Koblenz teaches: *moving said one or more ...points to the top of a loop process if they can be moved there without a problem occurring*; See Koblenz column 8 lines 23-28:

> Irreducible loops do not exhibit a loop top; however, all basic blocks in an irreducible loop that are reached by a forward control flow edge from a basic block outside the loop can be combined into a single summary loop top in constructing the tile tree. This summary node will dominate every basic block in the loop.

Also, in an analogous environment, Bak teaches: *transferring, from an interpreter process to a compiled code process, a method that is currently being executed for code that includes a plurality of transfer points at which program execution is transferred from the interpreter process to the compiled code process <via one of said transfer points>*. See column 2 lines 40-45:

> the hybrid virtual and native machine instructions may be easily transformed back to the original virtual machine instructions, and the flexibility of compiling only certain portions of a function into native machine instructions allows for better optimization of the execution of the function.

storing information for generating recalculation code for one or more specific transfer points See column 2 line 65 – column 3 line 1:

> A copy of a selected virtual machine instruction at a beginning of the portion of the function is **stored** and a back pointer to a location of the selected virtual machine instruction is also stored.

and performing a recalculation during a transfer process See column 3 lines 1-5:

> The selected virtual machine instruction is overwritten with a new virtual machine **instruction that specifies execution** of the native machine instructions so that the function includes both virtual and native machine instructions.

Bak is generally concerned with mixed execution of interpreted and compiled code sequences where the compiled code process is referred to as a "snippet", and transfer points to the compiled code process are indicated with a "go--native" instruction (column 6 lines 31-35 and column 7 lines 28-37).

Also in an analogous environment, Bacon teaches: *privatization* See page 395 Section 7.1.3:

> When a scalar is used within a loop solely as a scratch variable, each processor can be given a private copy so the use of the scalar need not involve any communication.

It would have been obvious to one of ordinary skill in the art at the time the invention was made to use Koblenz' moving of transfer points with Bacon's optimizations with Bak's mixed mode interpreter in Aho's code optimizer. One of ordinary skill would have been motivated to remove transfer points from a loop in order to make them reducible since many optimizations depend on reducibility (Aho page 607). Further, one would have been motivated to transfer the execution of an interpreted loop to natively compiled instructions since native code executes more quickly than interpreted code.

As per claim 2, the above rejection of claim 1 is incorporated. Aho does not expressly disclose choosing transfer points for transferring from interpreted mode to compiled mode execution.

However, Bak teaches *defining as a new transfer point, a point from said interpreter process to said compiled code process whereat, when said method that is currently being executed is replaced, the execution speed is increased compared with when said method is not replaced* (column 6 line 61 – column 7 line 5).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to use Bak's selection of transfer points in O'Brien's code optimizer.

One of ordinary skill would have been motivated to improve code so that a program will execute in less time.

As per claim 3, the above rejections of claims 1 and 2 are incorporated. Aho does not expressly disclose generating, storing, or employing information for transferring execution from interpreted to compiled execution.

However, Bak teaches:

generating information required to perform a transfer from said interpreter process to said compiled code process (column 7 lines 28-40); *and*

storing said generated information while correlating said generated information with said transfer points (column 7 lines 28-40 as cited above),

wherein, at said recalculation step, said information stored for said transfer points is employed (column 7 lines 63-67).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to use Bak's transfer information with O'Brien's code optimizer. One of ordinary skill would have been motivated to enable the transfer of interpreted execution to natively compiled execution, which is necessarily supported by information regarding the location of code, to increase the speed of a program.

As per claim 4, Aho does not expressly disclose a program storage device.

However, Bak teaches the use of a program storage device to hold program instructions (column 4 lines 46-50).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to use Bak's program storage device with O'Brien's code optimizer. One of ordinary skill would have been motivated to store copies of a program on media that enables the distribution of the program to colleagues or customers.

All further limitations have been addressed in the above rejection of claim 1.

- Claims 5, 6, 8, and 9 are rejected under 35 U.S.C. 103(a) as being unpatentable over Bak, in view of Koblenz and Aho.

In regard to claim 5, all limitations have been addressed in the above rejection of claim 1. It would have been obvious to one of ordinary skill in the art at the time the invention was made to use Koblenz' movement of transfer points, and Aho's optimization with Bak's transfer of execution. One of ordinary skill would have been motivated to make a loop reducible in order to better optimize it (Bak column 2 lines 24-29, Aho first paragraph of section 10.9 on page 660, and Koblenz column 8 lines 15-23).

In regard to claim 6, all limitations have been addressed in the above rejection of claim 1. It would have been obvious to one of ordinary skill in the art at the time the invention was made to use Aho's code copying with Bak's transfer process in order to facilitate the reducibility of a graph which would allow better optimization.

In regard to claim 8, all limitations have been addressed in the above rejections of claims 4 and 5.

In regard to claim 9, the above rejection of claim 8 is incorporated. All further limitations have been addressed in the above rejection of claim 1.

- Claims 7 and 10 are rejected under 35 U.S.C. 103(a) as being unpatentable over Aho in view of Bak.

In regard to claim 7, all limitations have been addressed in the above rejection of claim 1.

In regard to claim 10, all limitations have been addressed in the above rejection of claims 1 and 4.

(10) Response to Argument

Appellants provide two main arguments: 1) Moving Transfer Points to Top of Loop Process (see Brief, pages 10-12); and 2) Copying Code from Top of Loop Process (see Brief, pages 12-13). These arguments are addressed in turn below.

Premise 1. Moving Transfer Points to Top of Loop Process (see Brief, pages 10-12)

Appellants essentially argue that the prior art of record *Koblenz*, column 8:23-28, does not teach moving transfer points to the top of a loop process, since there is no motivation to combine the reference with prior art of record *Aho*. This argument is presented through several sub-arguments including:

a) *Koblenz*' summary loop top is created during construction of a tile tree, and is not concerned with the reducibility of a loop (see bottom of page 10 through top of page 11).

b) *Koblenz* is concerned with assigning physical registers, not loop optimization (see page 11, 2nd paragraph).

c) *Koblenz'* teaching of abstract flow graphs are not done for the comparable purpose and result of *Aho* (see bottom of page 11).

In response to Appellants' argument that there is no suggestion to combine the references, the examiner recognizes that obviousness can only be established by combining or modifying the teachings of the prior art to produce the claimed invention where there is some teaching, suggestion, or motivation to do so found either in the references themselves or in the knowledge generally available to one of ordinary skill in the art. See *In re Fine*, 837 F.2d 1071, 5 USPQ2d 1596 (Fed. Cir. 1988) and *In re Jones*, 958 F.2d 347, 21 USPQ2d 1941 (Fed. Cir. 1992). In this case, the motivation to combine comes from the references themselves, since Aho teaches that code optimizations benefit from reducible loops (see Aho page 607), and Koblenz teaches techniques for the transformation of irreducible loops into reducible loops (see Koblenz column 8 lines 23-28). Appellant's arguments appear to take the references out of context since both references are directed to compiling, and both are interested in optimization.

Regarding sub-argument *a)*, *Aho* discloses that loops are irreducible due to the presence of transfer points (e.g. see *Aho* Fig. 10.18, loop has two "headers," or transfer points, and is irreducible). *Koblenz* teaches a solution for moving transfer points out of a loop. In creating a "summary loop top," *Koblenz* moves the transfer points out of the body of the loop, thus

allowing further optimization as suggested by prior art of record *Aho* (see *Aho* pages 606-607, and page 660, e.g. "we can find optimization algorithms that run significantly faster on reducible flow graphs").

Regarding sub-argument *b)*, the assigning of physical registers is done using compilers, which is well within *Aho's* field of endeavor. *Aho* is concerned with all aspects of compiling, including optimization techniques. And from *Koblenz'* abstract, "The present invention provides methods for allocating physical registers **within a compiler phase to achieve efficient operation** of a target CPU" [emphasis added]. *Koblenz'* physical register assignment teaches compiler techniques directed to optimized operation of a CPU, and in the process teaches loop optimization. Clearly, one of ordinary skill in the art would look to *Koblenz* when compiling efficient code.

Finally, regarding sub-argument *c), Koblenz*' discussion of abstract flow graphs is indeed comparable to *Aho*, since *Aho* must utilize an abstract flow graph in order to make optimizations. *Koblenz* enhances a topic already present in *Aho*. Thus, *Koblenz* teaches a comparable purpose and result, and Appellants' arguments are not persuasive.

Premise 2. Copying Code from Top of Loop Process (see Brief, pages 12-13)

Appellants essentially argue (see page 12 paragraph 3) that *Aho* does not disclose entry points or transfer points, and thus "does not teach copying code represented by a set of (one or more) nodes, extending from the top of a loop to a point that post-dominates the top of the loop and a transfer point, to a location immediately preceding the loop" [emphasis in original]. It is noted that the claims are not directed to "entry points." Therefore, this aspect of the argument is

moot. Further, Appellant has provided a detailed description of transfer points on pages 5 and 6 of the Brief. The 12/30/03 Office action raised a question of enablement regarding such transfer points, since the originally filed disclosure only provided vague descriptions (e.g. page 4 line 4, page 8 paragraph 1, page 14 line 28). In response, Appellant provided a description similar to pages 5 and 6 of the Brief, in the 4/1/04 submission (see pages 5-7). In this submission, Appellants suggest that transfer points are well known in the art and disclosed in Section 10.4 (pp. 602-608) of *Aho*. Appellant currently admits that *Aho* discloses copying code, but particularly disputes the presence of transfer points in the cited portion of *Aho* (i.e. Fig. 10.49 on page 668). This is a marked departure from Appellants' 4/1/04 submission, which Appellants are now disputing. Note that Fig. 10.18 appearing on page 607 (appearing within Appellants' citation of support for transfer points) corresponds identically to Fig. 10.49(a) used in the rejection of the claims. Fig. 10.18 has been broadly interpreted according to Appellant's arguments as disclosing transfer points as edges that appear between the nodes of the figure. Appellant has not provided any reason why Fig. 10.49 does not similarly disclose transfer points within the context of *Aho*'s disclosure of transfer points in section 10.4, as admitted by Appellants.

Claims 5-10 (see pages 13-15)

All further arguments regarding claims 5-10 are based on previous arguments, addressed above.

(11) Related Proceeding(s) Appendix

No decision rendered by a court or the Board is identified by the examiner in the Related Appeals and Interferences section of this examiner's answer.

For the above reasons, it is believed that the rejections should be sustained.

Respectfully submitted,

/J. Derek Rutten/
Patent Examiner, Art Unit 2192 JDR

Conferees:

Tuan Dam, SPE AU 2192

TUAN DAM
SUPERVISORY PATENT EXAMINER

Wei Zhen, SPE AU 2191

WEI ZHEN
SUPERVISORY PATENT EXAMINER